```python
import gdown
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

Downloading the data

```python
customerCsv : str = "https://drive.google.com/file/d/1bu_--
mo79VdUG9oin4ybfFGRUSXAe-WE/view?usp=sharing"
productsCsv : str = "https://drive.google.com/file/d/1IKuDizVapw-
hyktwfpoAoaGtHtTNHfd0/view?usp=sharing"
transactionsCsv : str = "https://drive.google.com/file/d/1saEqdbBB-
vuk2hxoAf4TzDEsykdKlzbF/view?usp=sharing"

def download_csv(url : str, outputPath : str) -> None:
  try:
    gdown.download(url, outputPath, quiet=False, fuzzy=True)
    print(f"File downloaded successfully and saved to {outputPath}")
  except Exception as e:
    print(f"Failed to dowload the file {e}")

download_csv(customerCsv, "/content/")
download_csv(productsCsv, "/content/")
download_csv(transactionsCsv, "/content/")
```

```
Downloading...
From: https://drive.google.com/uc?id=1bu_--mo79VdUG9oin4ybfFGRUSXAe-WE
To: /content/Customers.csv
100%|██████████| 8.54k/8.54k [00:00<00:00, 7.77MB/s]

File downloaded successfully and saved to /content/

Downloading...
From: https://drive.google.com/uc?id=1IKuDizVapw-hyktwfpoAoaGtHtTNHfd0
To: /content/Products.csv
100%|██████████| 4.25k/4.25k [00:00<00:00, 3.04MB/s]

File downloaded successfully and saved to /content/

Downloading...
From: https://drive.google.com/uc?id=1saEqdbBB-vuk2hxoAf4TzDEsykdKlzbF
To: /content/Transactions.csv
100%|██████████| 54.7k/54.7k [00:00<00:00, 4.00MB/s]
```

```
File downloaded successfully and saved to /content/
```

Loading the dataset

```
dfCustomers = pd.read_csv("/content/Customers.csv")
dfTransactions = pd.read_csv("/content/Transactions.csv")
dfProducts = pd.read_csv("/content/Products.csv")

dfTransactions.head()
```

```
{"summary":"{\n  \"name\": \"dfTransactions\",\n  \"rows\": 1000,\n
\"fields\": [\n    {\n      \"column\": \"TransactionID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"T00677\",\n          \"T00790\",\n          \"T00907\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CustomerID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 199,\n        \"samples\": [\n
\"C0135\",\n          \"C0109\",\n          \"C0048\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ProductID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 100,\n        \"samples\": [\n
\"P082\",\n          \"P052\",\n          \"P035\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TransactionDate\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"2024-03-05 23:39:40\",\n          \"2024-08-13 23:52:47\",\n
\"2024-02-15 17:18:56\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Quantity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 4,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          4,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TotalValue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
493.14447754793144,\n        \"min\": 16.08,\n        \"max\":
1991.04,\n        \"num_unique_values\": 369,\n        \"samples\": [\
n          1789.36,\n          681.78,\n          580.34\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 140.73638962578207,\
n        \"min\": 16.08,\n        \"max\": 497.76,\n
```

\"num_unique_values\": 100,\n   \"samples\": [\n   55.99,\
n   354.81,\n   30.59\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"dfTransactions"}

```
dfProducts.head()
```

{"summary":"{\n \"name\": \"dfProducts\",\n \"rows\": 100,\n
\"fields\": [\n   {\n   \"column\": \"ProductID\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 100,\n   \"samples\": [\n
\"P084\",\n   \"P054\",\n   \"P071\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"ProductName\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 66,\n   \"samples\": [\n
\"ComfortLiving Laptop\",\n   \"BookWorld Running Shoes\",\n
\"ActiveWear Biography\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Category\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 4,\n   \"samples\":
[\n   \"Electronics\",\n   \"Clothing\",\n
\"Books\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Price\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 143.21938309125758,\n   \"min\": 16.08,\n
\"max\": 497.76,\n   \"num_unique_values\": 100,\n
\"samples\": [\n   337.91,\n   57.3,\n   127.36\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"dfProducts"}

```
dfCustomers.head()
```

{"summary":"{\n \"name\": \"dfCustomers\",\n \"rows\": 200,\n
\"fields\": [\n   {\n   \"column\": \"CustomerID\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 200,\n   \"samples\": [\n
\"C0096\",\n   \"C0016\",\n   \"C0031\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"CustomerName\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 200,\n   \"samples\": [\n
\"Benjamin Mcclure\",\n   \"Emily Woods\",\n   \"Tina
Miller\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Region\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 4,\n   \"samples\":
[\n   \"Asia\",\n   \"Europe\",\n   \"South
America\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"SignupDate\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 179,\n        \"samples\":
[\n          \"2022-04-07\",\n          \"2023-12-05\",\n
\"2022-03-15\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"dfCustomers"}
```

```python
transactions_products = pd.merge(dfTransactions, dfProducts,
on='ProductID', how='left')
df = pd.merge(transactions_products, dfCustomers, on='CustomerID',
how='left')
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"TransactionID\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
1000,\n        \"samples\": [\n          \"T00677\",\n
\"T00790\",\n          \"T00907\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CustomerID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 199,\n        \"samples\": [\n
\"C0135\",\n          \"C0109\",\n          \"C0048\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ProductID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 100,\n        \"samples\": [\n
\"P082\",\n          \"P052\",\n          \"P035\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TransactionDate\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"2024-03-05 23:39:40\",\n          \"2024-08-13 23:52:47\",\n
\"2024-02-15 17:18:56\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Quantity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 4,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          4,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TotalValue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
493.14447754793144,\n        \"min\": 16.08,\n        \"max\":
1991.04,\n        \"num_unique_values\": 369,\n        \"samples\": [\
n          1789.36,\n          681.78,\n          580.34\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price_x\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
140.73638962578207,\n        \"min\": 16.08,\n        \"max\":
497.76,\n        \"num_unique_values\": 100,\n        \"samples\": [\n
55.99,\n          354.81,\n          30.59\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ProductName\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 66,\n        \"samples\": [\n
\"ActiveWear Jacket\",\n          \"BookWorld Bluetooth Speaker\",\n
\"ComfortLiving Bluetooth Speaker\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Category\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"Clothing\",\n          \"Home
Decor\",\n          \"Electronics\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price_y\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
140.73638962578207,\n        \"min\": 16.08,\n        \"max\":
497.76,\n        \"num_unique_values\": 100,\n        \"samples\": [\n
55.99,\n          354.81,\n          30.59\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CustomerName\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 199,\n        \"samples\": [\n          \"Toni
Weaver\",\n          \"Abigail Jones\",\n          \"Matthew Park\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Region\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"Asia\",\n          \"North
America\",\n          \"Europe\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SignupDate\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 178,\n        \"samples\": [\n          \"2023-
06-11\",\n          \"2023-09-27\",\n          \"2022-02-10\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```python
df.isnull().sum()
```

```
TransactionID      0
CustomerID         0
ProductID          0
TransactionDate    0
Quantity           0
TotalValue         0
Price_x            0
ProductName        0
Category           0
Price_y            0
CustomerName       0
Region             0
```

```
SignupDate         0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

Data Cleaning and Preprocessing

```
if "Unnamed: 0" in df.columns:
  df = df.drop(columns=["Unnamed: 0"])

df["TransactionDate"] = pd.to_datetime(df["TransactionDate"],
errors="coerce")
df["SignupDate"] = pd.to_datetime(df["SignupDate"], errors = "coerce")
df['TransactionMonth'] = df['TransactionDate'].dt.to_period('M')

duplicates = df.duplicated().sum()
dfCleaned = df.drop_duplicates()

print(f"Duplicates removed : {duplicates}")
print(f"Cleaned Shape : {dfCleaned.shape}")
print(f"Date Conversion Success: {dfCleaned[['TransactionDate',
'SignupDate']].dtypes.to_dict()}")
```

```
Duplicates removed : 0
Cleaned Shape : (1000, 14)
Date Conversion Success: {'TransactionDate': dtype('<M8[ns]'),
'SignupDate': dtype('<M8[ns]')}
```

Discriptive Statistics

```
dfCleaned.describe()
```

```
{"summary":"{\n  \"name\": \"dfCleaned\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"TransactionDate\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"1970-01-01 00:00:00.000001\",\n        \"max\": \"2024-12-28
11:00:00\",\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          \"1000\",\n          \"2024-06-23 15:33:02.768999936\",\n
\"2024-09-19 14:19:57\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Quantity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 352.66353426013046,\n        \"min\":
1.0,\n        \"max\": 1000.0,\n        \"num_unique_values\": 7,\n
\"samples\": [\n          1000.0,\n          2.537,\n          4.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
```

}\n   },\n   {\n   \"column\": \"TotalValue\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
598.9454831884048,\n   \"min\": 16.08,\n   \"max\":
1991.04,\n   \"num_unique_values\": 8,\n   \"samples\": [\n
689.9955600000001,\n   1011.66,\n   1000.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Price_x\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
305.16091561989646,\n   \"min\": 16.08,\n   \"max\":
1000.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n
272.55407,\n   404.4,\n   1000.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Price_y\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
305.16091561989646,\n   \"min\": 16.08,\n   \"max\":
1000.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n
272.55407,\n   404.4,\n   1000.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"SignupDate\",\n
\"properties\": {\n   \"dtype\": \"date\",\n   \"min\":
\"1970-01-01 00:00:00.000001\",\n   \"max\": \"2024-12-28
00:00:00\",\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   \"1000\",\n   \"2023-07-09 02:49:55.199999744\",\n
\"2024-04-12 00:00:00\"\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe"}

```
numericDf = df[["Quantity", "TotalValue", "Price_x", "Price_y"]]
numericDf.head()
```

{"summary":"{\n \"name\": \"numericDf\",\n \"rows\": 1000,\n
\"fields\": [\n   {\n   \"column\": \"Quantity\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1,\n   \"min\": 1,\n   \"max\": 4,\n
\"num_unique_values\": 4,\n   \"samples\": [\n   2,\n
4,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"TotalValue\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 493.14447754793144,\n   \"min\":
16.08,\n   \"max\": 1991.04,\n   \"num_unique_values\":
369,\n   \"samples\": [\n   1789.36,\n   681.78,\n
580.34\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Price_x\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 140.73638962578207,\n   \"min\": 16.08,\n
\"max\": 497.76,\n   \"num_unique_values\": 100,\n
\"samples\": [\n   55.99,\n   354.81,\n   30.59\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Price_y\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":

140.73638962578207,\n   \"min\": 16.08,\n   \"max\":
497.76,\n   \"num_unique_values\": 100,\n   \"samples\": [\n
55.99,\n   354.81,\n   30.59\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"numericDf"}

```
numericDf.var()
```

```
Quantity            1.249881
TotalValue     243191.475736
Price_x         19806.731365
Price_y         19806.731365
dtype: float64
```

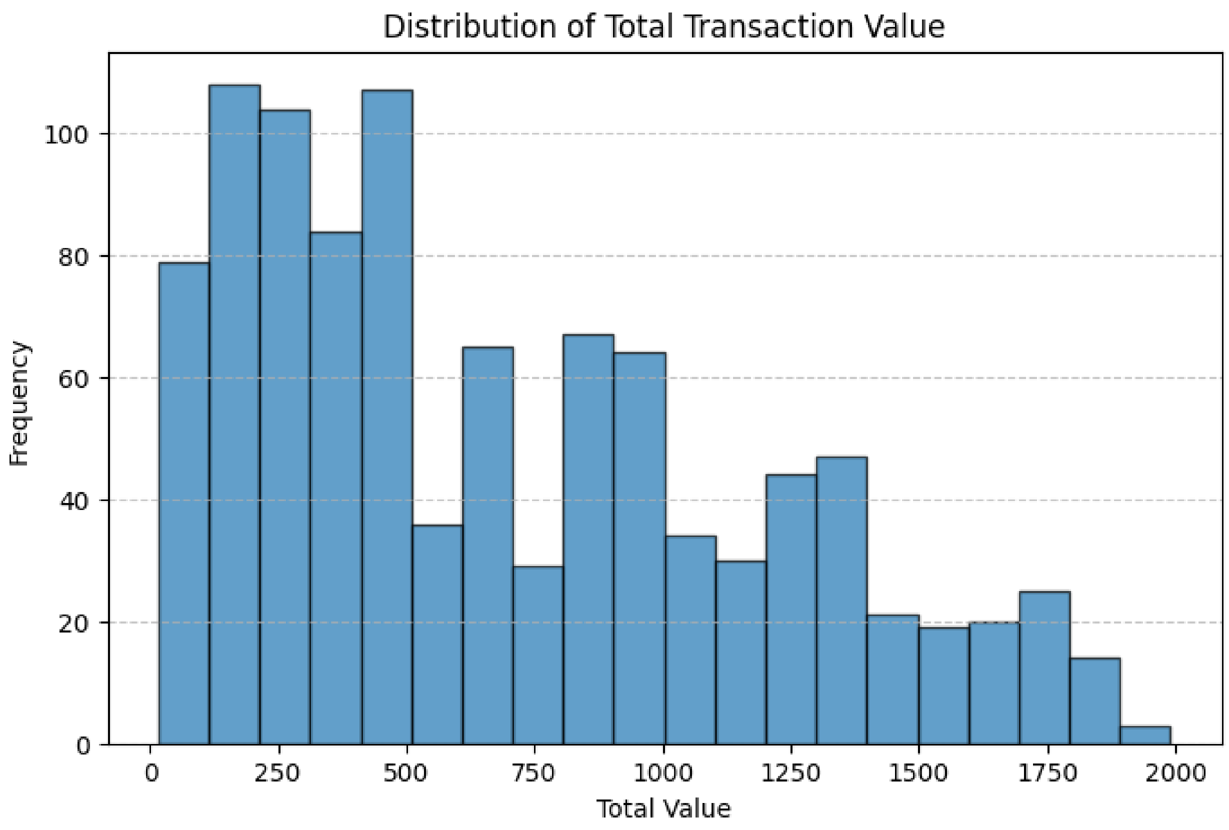
```
range = numericDf.max() - numericDf.min()
```

```
Quantity         3.00
TotalValue    1974.96
Price_x        481.68
Price_y        481.68
dtype: float64
```

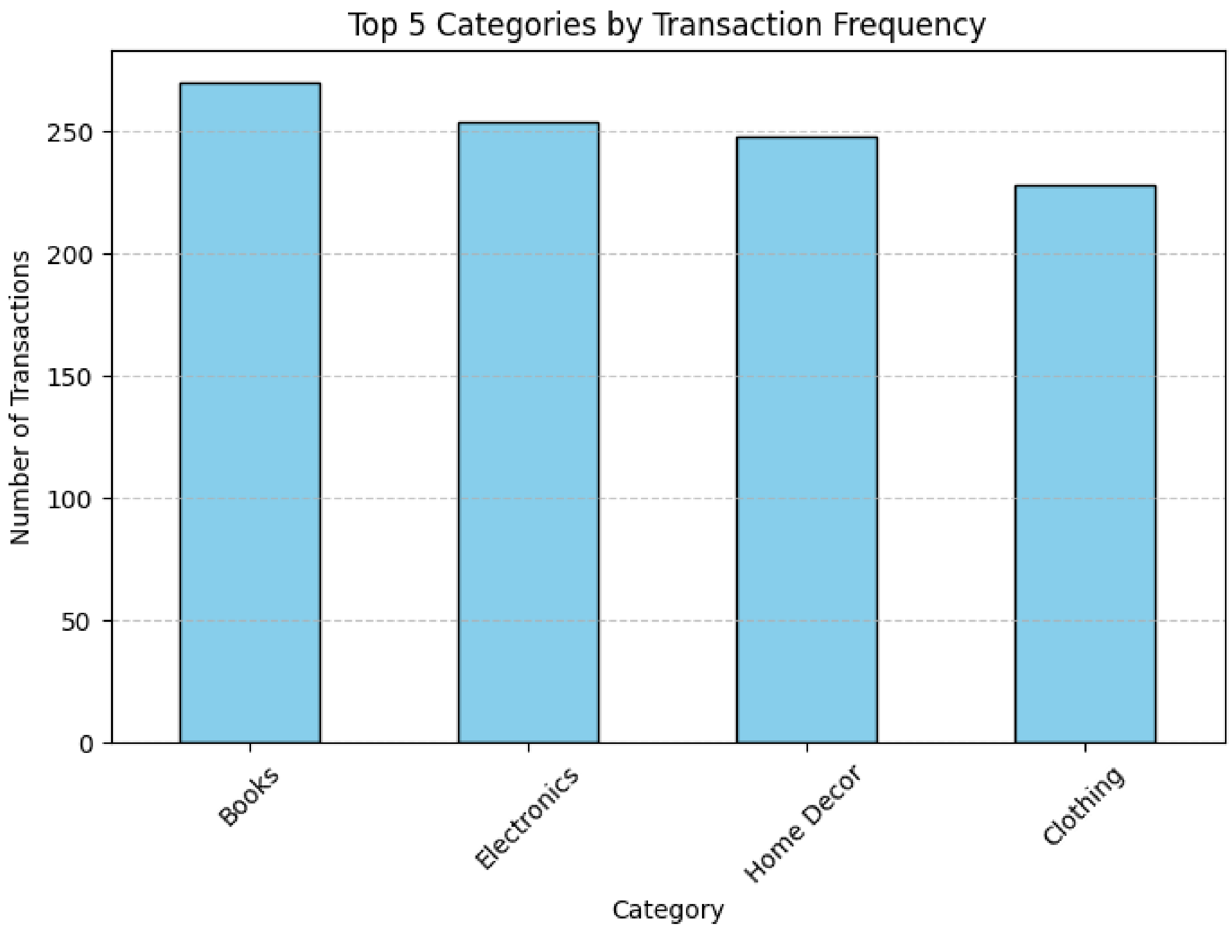
```
q1 = numericDf.quantile(0.25)
q3 = numericDf.quantile(0.75)
iqr = q3 - q1
iqr
```

```
Quantity        2.000
TotalValue    716.365
Price_x       256.450
Price_y       256.450
dtype: float64
```

```
plt.figure(figsize=(8, 5))
plt.hist(dfCleaned['TotalValue'], bins=20, edgecolor='k', alpha=0.7)
plt.title('Distribution of Total Transaction Value')
plt.xlabel('Total Value')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

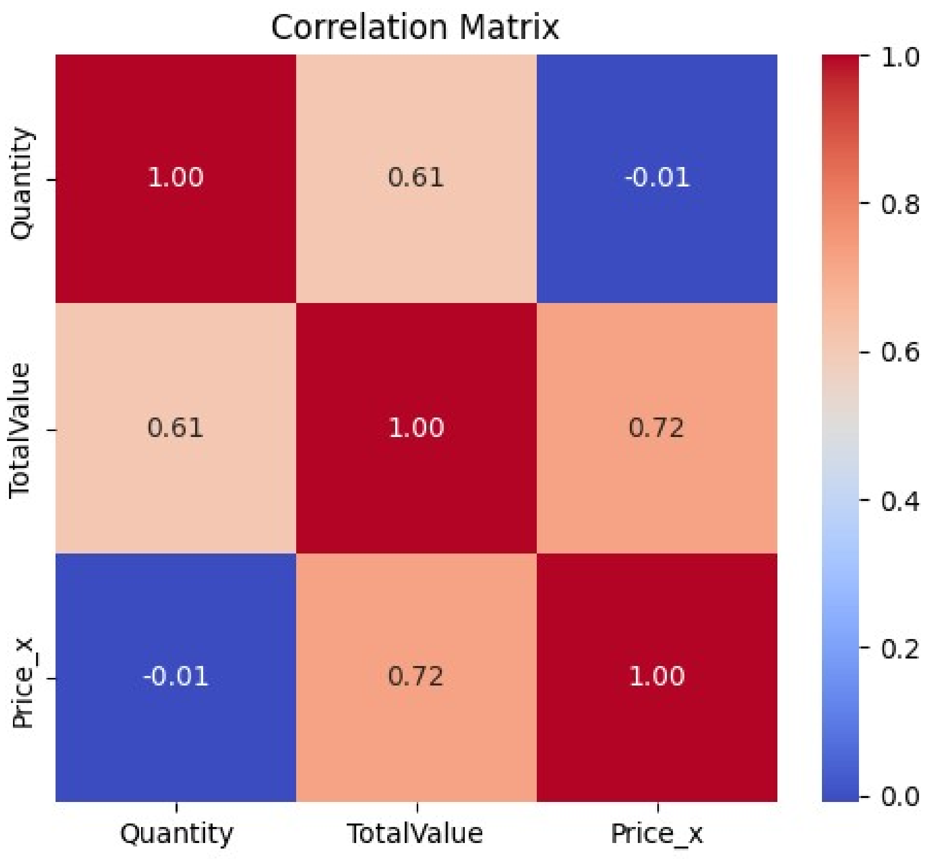


```python
top_categories = dfCleaned['Category'].value_counts().head(5)

plt.figure(figsize=(8, 5))
top_categories.plot(kind='bar', color='skyblue', edgecolor='k')
plt.title('Top 5 Categories by Transaction Frequency')
plt.xlabel('Category')
plt.ylabel('Number of Transactions')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Trend and Correlation Analysis

```
correlation_matrix = dfCleaned[['Quantity', 'TotalValue',
'Price_x']].corr()
plt.figure(figsize=(6, 5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

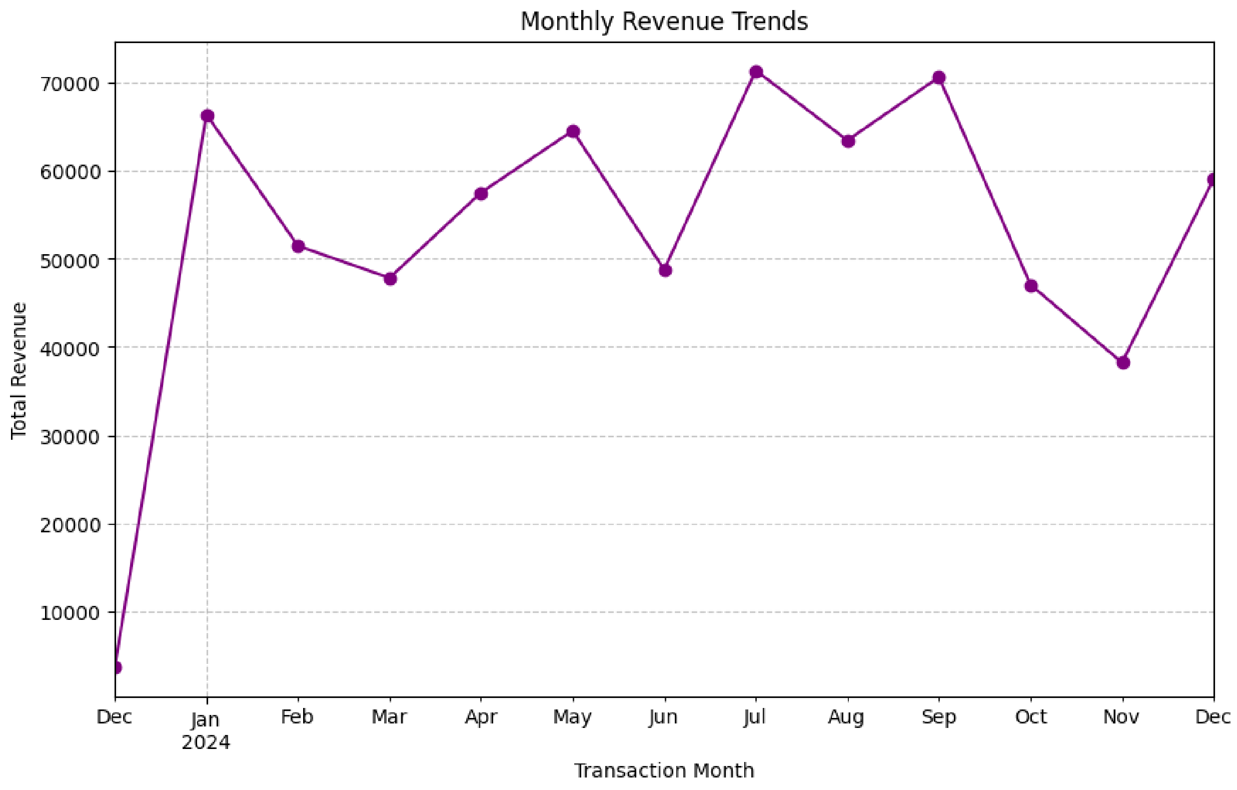


```
dfCleaned.head()
```

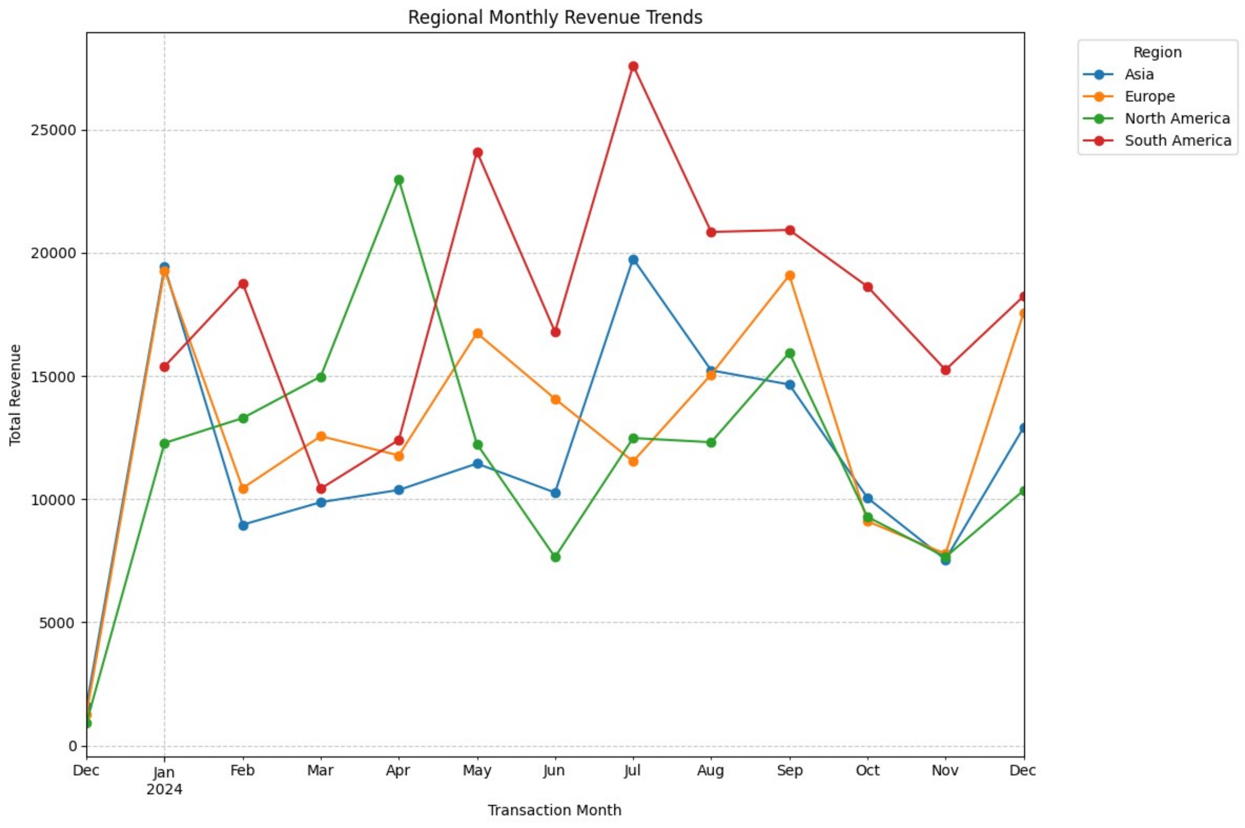
```
{"summary":"{\n  \"name\": \"dfCleaned\",\n  \"rows\": 1000,\n
\"fields\": [\n    {\n      \"column\": \"TransactionID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"T00677\",\n          \"T00790\",\n          \"T00907\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CustomerID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 199,\n        \"samples\": [\n
\"C0135\",\n          \"C0109\",\n          \"C0048\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ProductID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 100,\n        \"samples\": [\n
\"P082\",\n          \"P052\",\n          \"P035\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TransactionDate\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2023-12-30 15:29:12\",\n        \"max\": \"2024-12-28 11:00:00\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"2024-03-05 23:39:40\",\n          \"2024-08-13 23:52:47\",\n
\"2024-02-15 17:18:56\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Quantity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 4,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          4,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TotalValue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
493.14447754793144,\n        \"min\": 16.08,\n        \"max\":
1991.04,\n        \"num_unique_values\": 369,\n        \"samples\": [\
n          1789.36,\n          681.78,\n          580.34\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price_x\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
140.73638962578207,\n        \"min\": 16.08,\n        \"max\":
497.76,\n        \"num_unique_values\": 100,\n        \"samples\": [\n
55.99,\n          354.81,\n          30.59\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ProductName\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 66,\n        \"samples\": [\n
\"ActiveWear Jacket\",\n          \"BookWorld Bluetooth Speaker\",\n
\"ComfortLiving Bluetooth Speaker\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Category\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"Clothing\",\n          \"Home
Decor\",\n          \"Electronics\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price_y\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
140.73638962578207,\n        \"min\": 16.08,\n        \"max\":
497.76,\n        \"num_unique_values\": 100,\n        \"samples\": [\n
55.99,\n          354.81,\n          30.59\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CustomerName\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 199,\n        \"samples\": [\n          \"Toni
Weaver\",\n          \"Abigail Jones\",\n          \"Matthew Park\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Region\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"Asia\",\n          \"North
America\",\n          \"Europe\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SignupDate\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2022-01-22 00:00:00\",\n        \"max\": \"2024-12-28 00:00:00\",\n
```

\"num_unique_values\": 178,\n   \"samples\": [\n   \"2023-06-11 00:00:00\",\n   \"2023-09-27 00:00:00\",\n   \"2022-02-10 00:00:00\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"dfCleaned"}

```
monthly_revenue_trend = dfCleaned.groupby('TransactionMonth')
['TotalValue'].sum()

plt.figure(figsize=(10, 6))
monthly_revenue_trend.plot(kind='line', marker='o', linestyle='-',
color='purple')
plt.title('Monthly Revenue Trends')
plt.xlabel('Transaction Month')
plt.ylabel('Total Revenue')
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.show()
```

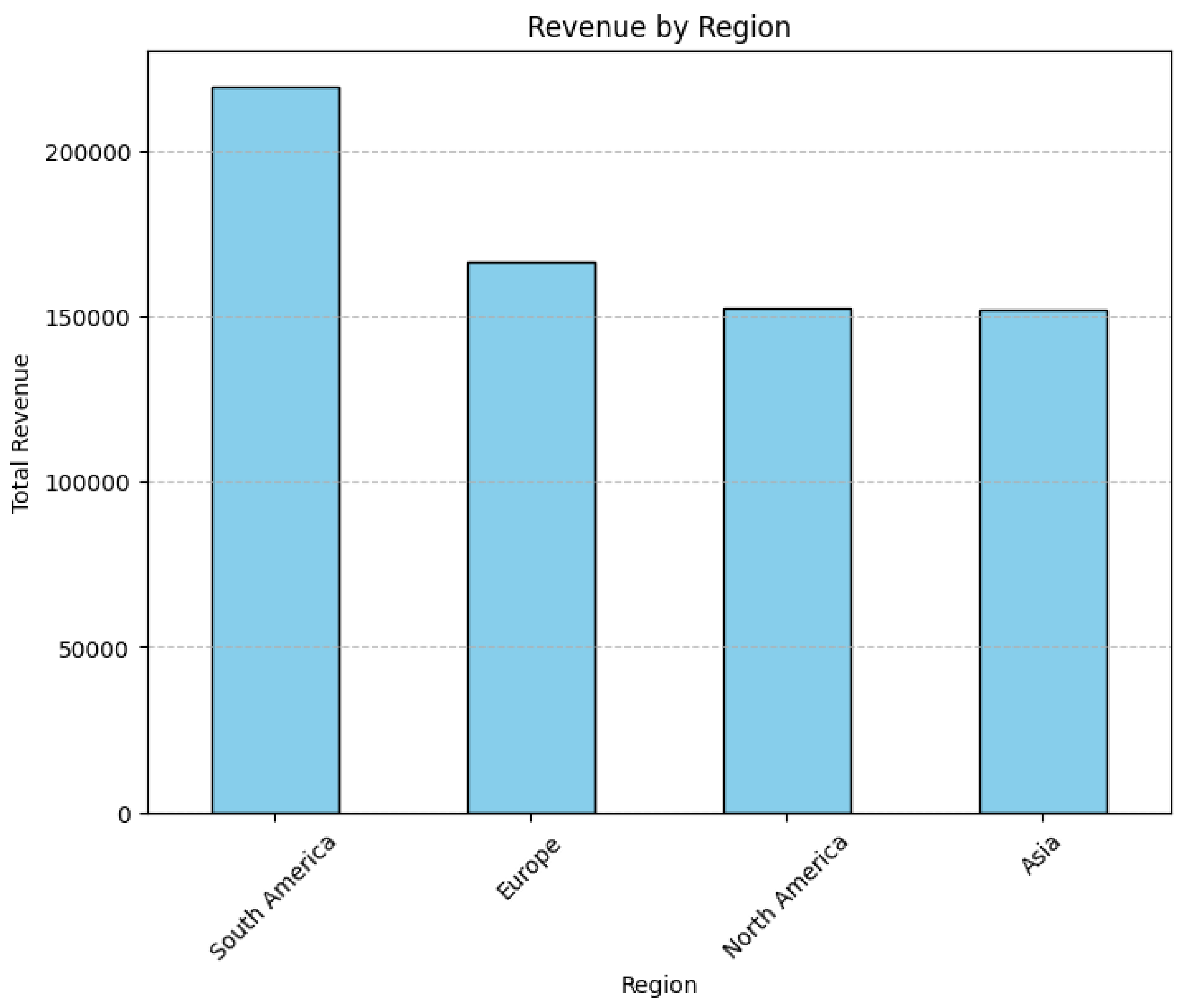


```
regional_revenue_trend = dfCleaned.groupby(['Region',
'TransactionMonth'])['TotalValue'].sum().unstack()

regional_revenue_trend.T.plot(kind='line', figsize=(12, 8),
marker='o')
plt.title('Regional Monthly Revenue Trends')
plt.xlabel('Transaction Month')
plt.ylabel('Total Revenue')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

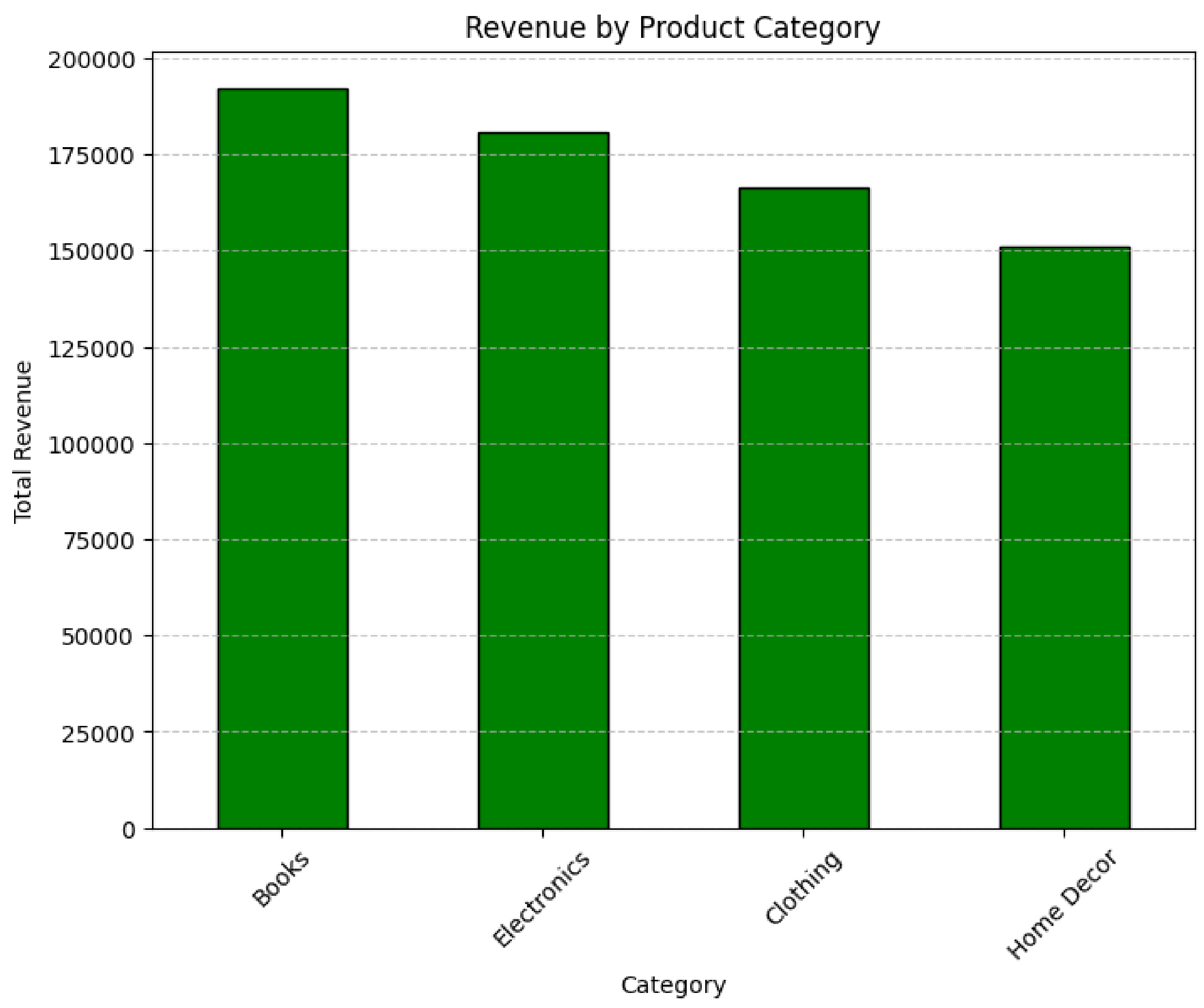


Segmentation and Group Analysis

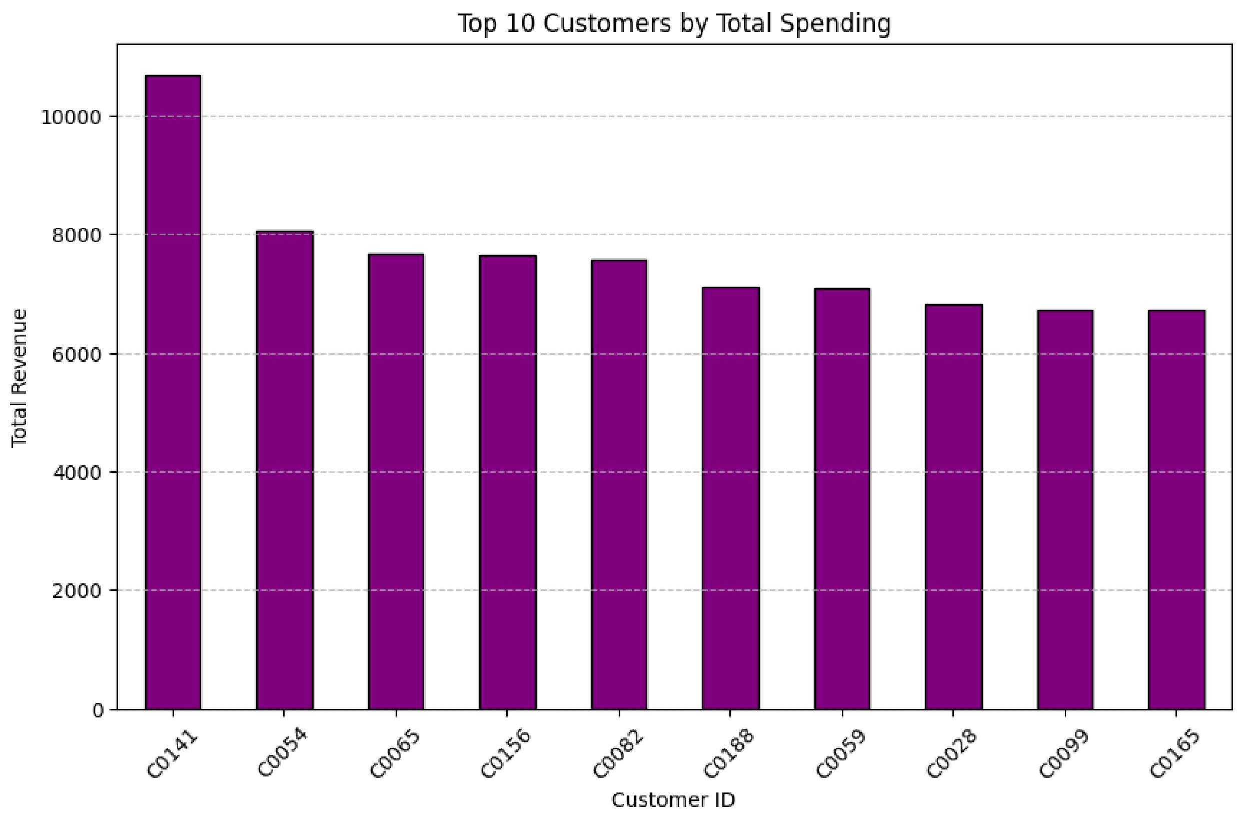
```
region_revenue = dfCleaned.groupby('Region')
['TotalValue'].sum().sort_values(ascending=False)
region_transaction_count = dfCleaned['Region'].value_counts()

plt.figure(figsize=(8, 6))
region_revenue.plot(kind='bar', color='skyblue', edgecolor='k')
plt.title('Revenue by Region')
plt.xlabel('Region')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



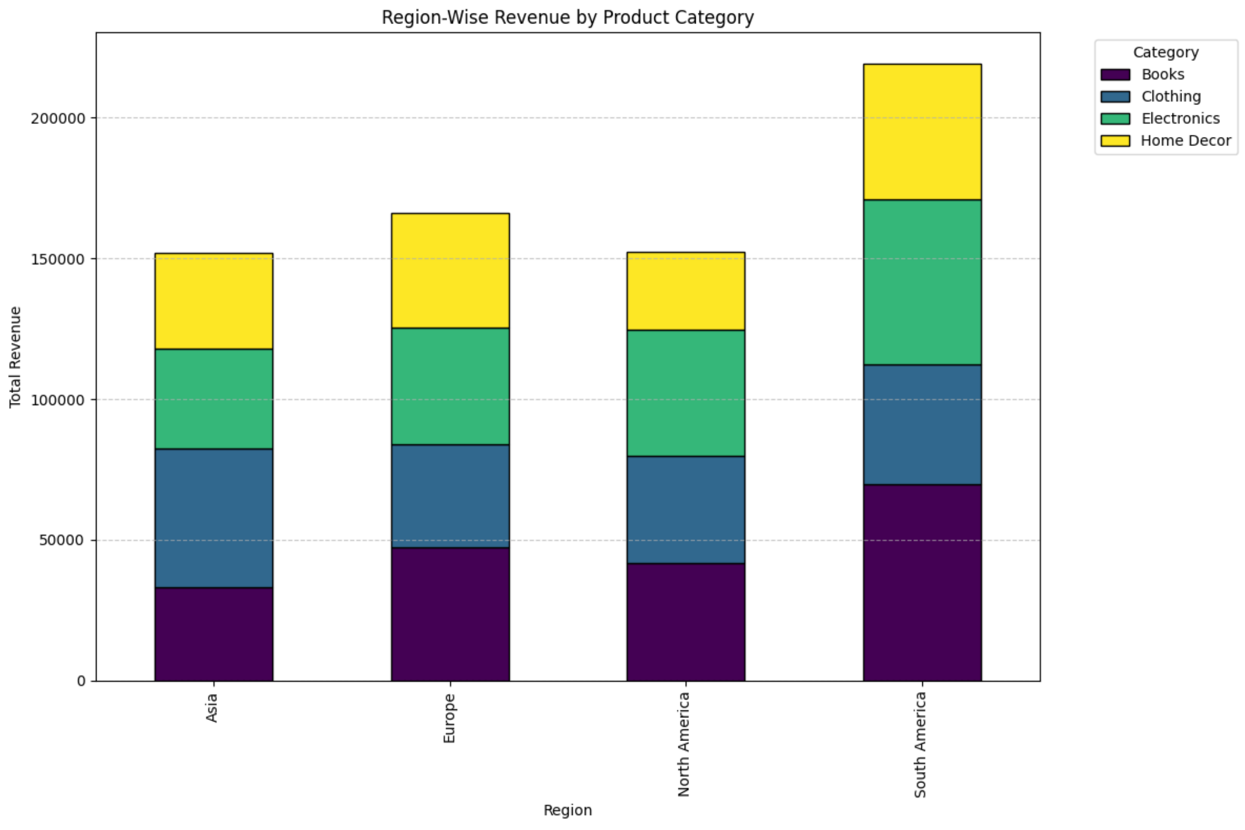
```
category_revenue = dfCleaned.groupby('Category')
['TotalValue'].sum().sort_values(ascending=False)
category_transaction_count = dfCleaned['Category'].value_counts()

plt.figure(figsize=(8, 6))
category_revenue.plot(kind='bar', color='green', edgecolor='k')
plt.title('Revenue by Product Category')
plt.xlabel('Category')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
customer_spending = dfCleaned.groupby('CustomerID')
['TotalValue'].sum().sort_values(ascending=False).head(10)

# Visualize top customers
plt.figure(figsize=(10, 6))
customer_spending.plot(kind='bar', color='purple', edgecolor='k')
plt.title('Top 10 Customers by Total Spending')
plt.xlabel('Customer ID')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



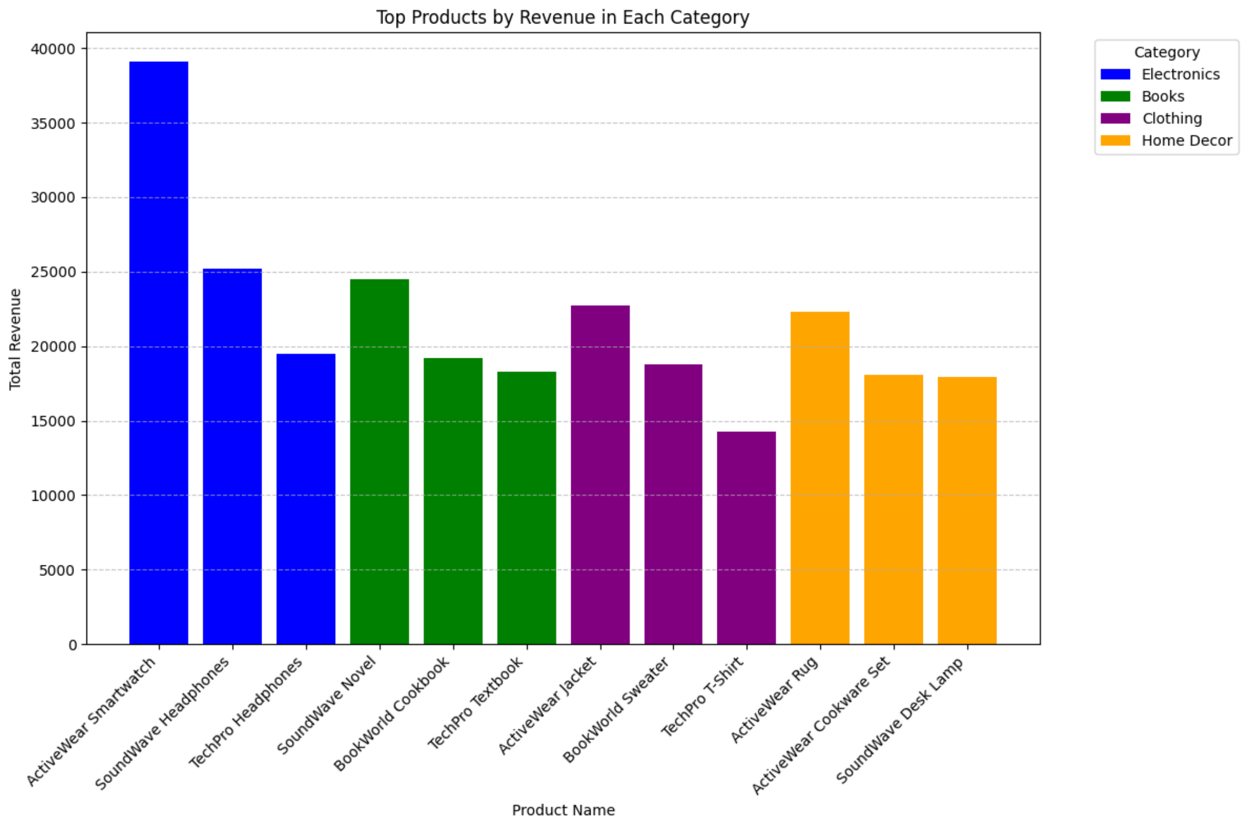
```
region_revenue, category_revenue, customer_spending

(Region
 South America    219352.56
 Europe           166254.63
 North America    152313.40
 Asia             152074.97
 Name: TotalValue, dtype: float64,
 Category
 Books          192147.47
 Electronics    180783.50
 Clothing       166170.66
 Home Decor     150893.93
 Name: TotalValue, dtype: float64,
 CustomerID
 C0141    10673.87
 C0054     8040.39
 C0065     7663.70
 C0156     7634.45
 C0082     7572.91
 C0188     7111.32
 C0059     7073.28
 C0028     6819.57
 C0099     6715.72
 C0165     6708.10
 Name: TotalValue, dtype: float64)

region_category_revenue = dfCleaned.groupby(['Region', 'Category'])
['TotalValue'].sum().unstack()

plt.figure(figsize=(12, 8))
region_category_revenue.plot(kind='bar', stacked=True, figsize=(12,
8), colormap='viridis', edgecolor='k')
plt.title('Region-Wise Revenue by Product Category')
plt.xlabel('Region')
plt.ylabel('Total Revenue')
plt.legend(title='Category', bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

region_category_revenue

<Figure size 1200x800 with 0 Axes>
```



{"summary":"{\n \"name\": \"region_category_revenue\",\n \"rows\":
4,\n \"fields\": [\n   {\n   \"column\": \"Region\",\n

```
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"Europe\",\n          \"South America\",\n          \"Asia\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Books\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
15632.945490300019,\n        \"min\": 33119.03,\n        \"max\":
69752.03,\n        \"num_unique_values\": 4,\n        \"samples\": [\n
47464.42,\n          69752.03,\n          33119.03\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Clothing\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
5808.244050413745,\n        \"min\": 36335.59,\n        \"max\":
49353.09,\n        \"num_unique_values\": 4,\n        \"samples\": [\n
36335.59,\n          42443.49,\n          49353.09\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Electronics\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
9844.345273765035,\n        \"min\": 35658.15,\n        \"max\":
58846.32,\n        \"num_unique_values\": 4,\n        \"samples\": [\n
41562.6,\n          58846.32,\n          35658.15\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Home Decor\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8868.461052096827,\n        \"min\": 27746.489999999998,\n
\"max\": 48310.72,\n        \"num_unique_values\": 4,\n
\"samples\": [\n          40892.02,\n          48310.72,\n
33944.7\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"region_category_revenue"}
```

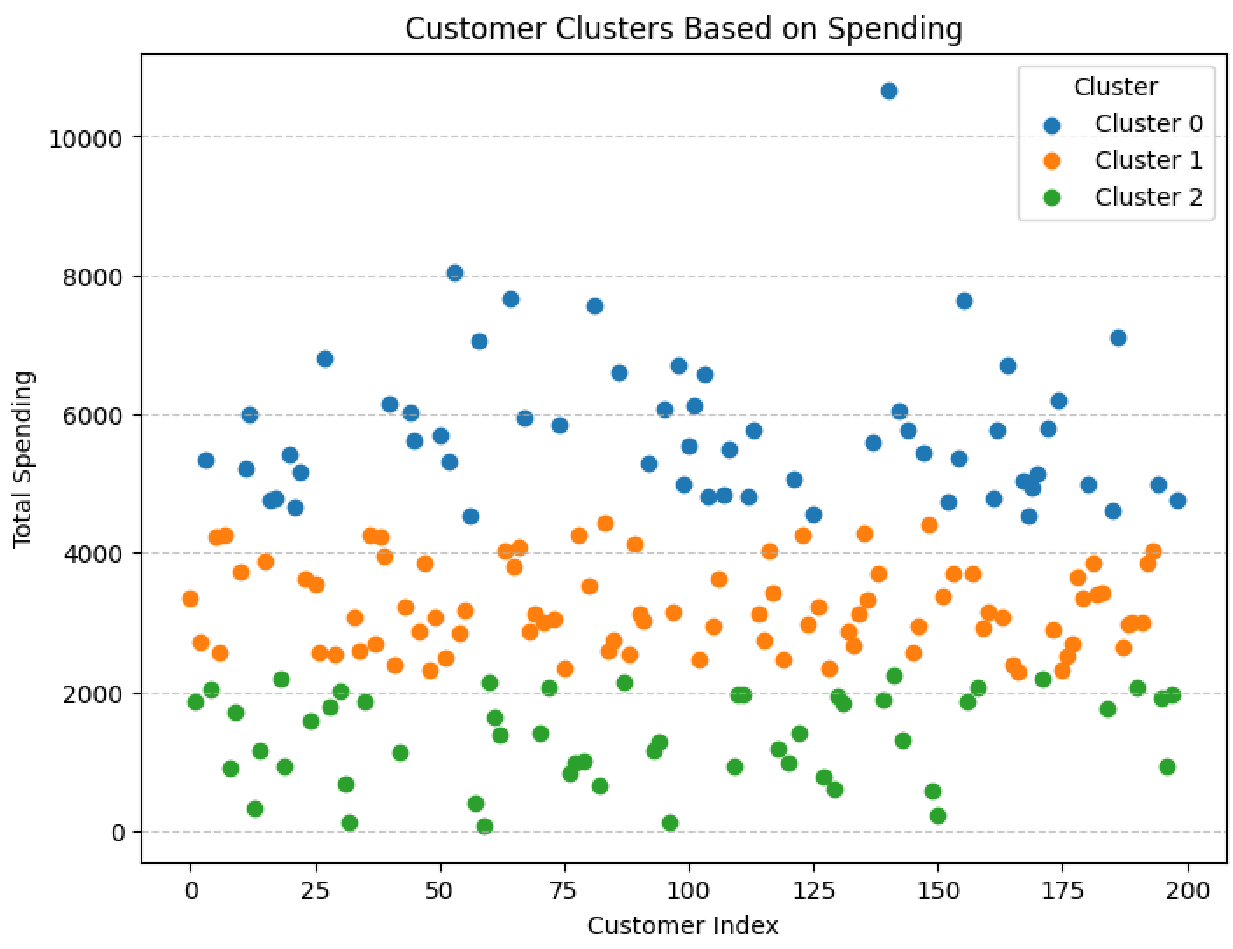
```
top_products_by_category = dfCleaned.groupby(['Category',
'ProductName'])['TotalValue'].sum().sort_values(ascending=False)
top_products_by_category =
top_products_by_category.groupby('Category').head(3).reset_index()

plt.figure(figsize=(12, 8))
categories = top_products_by_category['Category'].unique()
colors = ['blue', 'green', 'purple', 'orange']

for i, category in enumerate(categories):
    subset =
top_products_by_category[top_products_by_category['Category'] ==
category]
    plt.bar(subset['ProductName'], subset['TotalValue'],
color=colors[i % len(colors)], label=category)

plt.title('Top Products by Revenue in Each Category')
plt.xlabel('Product Name')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.legend(title='Category', bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

top_products_by_category
```



```
{"summary":"{\n  \"name\": \"top_products_by_category\",\n  \"rows\":
12,\n  \"fields\": [\n    {\n      \"column\": \"Category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"Books\",\n          \"Home Decor\",\n          \"Electronics\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"ProductName\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 12,\n        \"samples\": [\n
\"SoundWave Desk Lamp\",\n          \"ActiveWear Cookware Set\",\n
\"ActiveWear Smartwatch\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"TotalValue\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 6312.6427448531995,\n
\"min\": 14264.14,\n        \"max\": 39096.97,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
17920.1,\n          18083.73,\n          39096.97\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"top_products_by_category"}
```

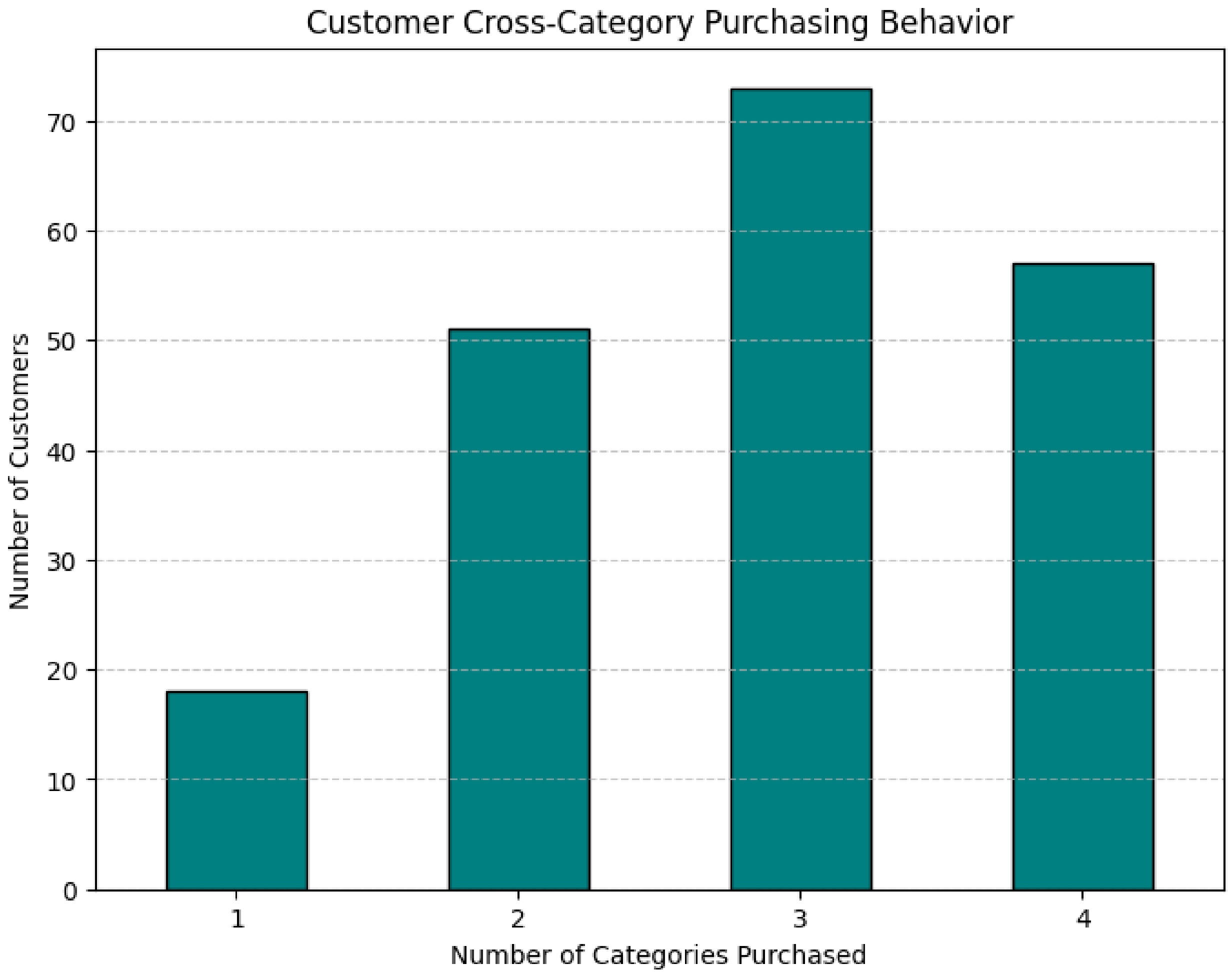
Lol

```
customer_spending = dfCleaned.groupby('CustomerID')
['TotalValue'].sum().reset_index()

scaler = StandardScaler()
customer_spending_scaled =
scaler.fit_transform(customer_spending[['TotalValue']])

kmeans = KMeans(n_clusters=3, random_state=42)
customer_spending['Cluster'] =
kmeans.fit_predict(customer_spending_scaled)

cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)

assert isinstance(customer_spending, pd.DataFrame), "customer_spending
is not a DataFrame."

plt.figure(figsize=(8, 6))
for cluster in [0, 1, 2]:
    cluster_data = customer_spending[customer_spending['Cluster'] ==
cluster]
    plt.scatter(
        cluster_data.index,
        cluster_data['TotalValue'],
        label=f'Cluster {cluster}'
    )

plt.title('Customer Clusters Based on Spending')
plt.xlabel('Customer Index')
plt.ylabel('Total Spending')
plt.legend(title='Cluster')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# Cluster Summary
cluster_summary = customer_spending.groupby('Cluster')
['TotalValue'].agg(['mean', 'min', 'max', 'count'])
cluster_centers, cluster_summary
```



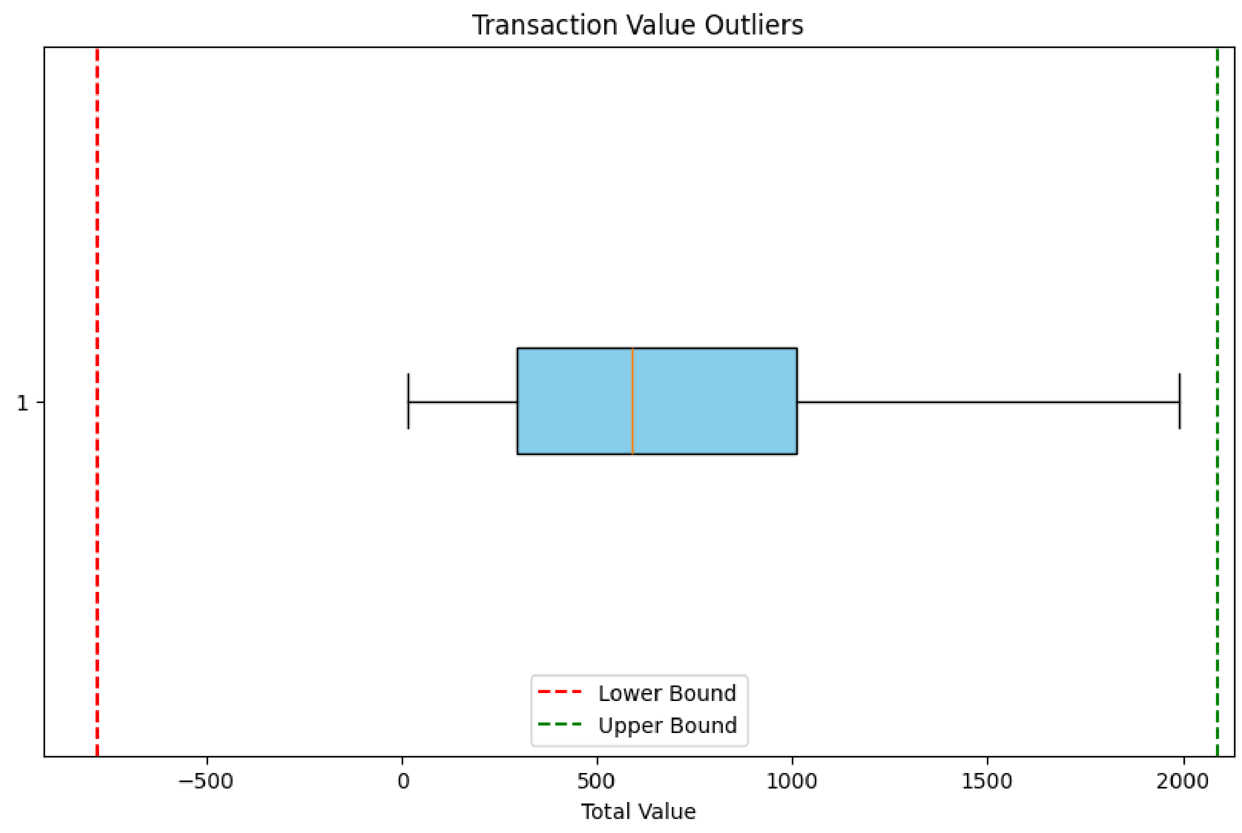
```
(array([[5754.03551724],
        [3227.22295455],
        [1363.50716981]]),
                     mean       min        max  count
 Cluster
 0        5754.035517  4533.32  10673.87     58
 1        3227.222955  2300.42   4441.10     88
 2        1363.507170    82.36   2239.04     53)

category_purchases = dfCleaned.groupby(['CustomerID', 'Category'])
['TotalValue'].sum().unstack().fillna(0)

# Add a column for total categories purchased
category_purchases['CategoryCount'] = (category_purchases >
0).sum(axis=1)

# Analyze cross-category trends
category_trend_summary =
category_purchases['CategoryCount'].value_counts()

# Visualize cross-category purchasing behavior
```

```
plt.figure(figsize=(8, 6))
category_trend_summary.sort_index().plot(kind='bar', color='teal',
edgecolor='k')
plt.title('Customer Cross-Category Purchasing Behavior')
plt.xlabel('Number of Categories Purchased')
plt.ylabel('Number of Customers')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=0)
plt.show()

# Return customers who purchase across all categories
all_category_customers =
category_purchases[category_purchases['CategoryCount'] ==
category_purchases.shape[1]]
category_trend_summary, all_category_customers
```



```
(CategoryCount
 3    73
 4    57
 2    51
 1    18
 Name: count, dtype: int64,
 Empty DataFrame
 Columns: [Books, Clothing, Electronics, Home Decor, CategoryCount]
 Index: [])
```

```
q1 = dfCleaned['TotalValue'].quantile(0.25)
q3 = dfCleaned['TotalValue'].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

# Identify outliers
outliers = dfCleaned[(dfCleaned['TotalValue'] < lower_bound) |
(dfCleaned['TotalValue'] > upper_bound)]

# Visualize distribution with outliers
plt.figure(figsize=(10, 6))
plt.boxplot(dfCleaned['TotalValue'], vert=False, patch_artist=True,
boxprops=dict(facecolor='skyblue'))
plt.axvline(lower_bound, color='red', linestyle='--', label='Lower
Bound')
plt.axvline(upper_bound, color='green', linestyle='--', label='Upper
Bound')
plt.title('Transaction Value Outliers')
plt.xlabel('Total Value')
plt.legend()
plt.show()

# Outlier details
outliers_summary = {
    "Outlier Count": len(outliers),
    "Highest Outlier": outliers['TotalValue'].max(),
    "Lowest Outlier": outliers['TotalValue'].min(),
    "Outlier Percentage": len(outliers) / len(dfCleaned) * 100,
}
outliers_summary, outliers.head()
```



```
({'Outlier Count': 0,
  'Highest Outlier': nan,
  'Lowest Outlier': nan,
  'Outlier Percentage': 0.0},
 Empty DataFrame
 Columns: [TransactionID, CustomerID, ProductID, TransactionDate,
Quantity, TotalValue, Price_x, ProductName, Category, Price_y,
CustomerName, Region, SignupDate, TransactionMonth]
 Index: [])
```